APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/665,305	09/22/2003	Thomas Goering	11884-400201	5450

53000 7590 04/03/2008
KENYON & KENYON LLP
1500 K STREET N.W.
WASHINGTON, DC 20005

EXAMINER	
DESAI, RACHNA SINGH	

ART UNIT	PAPER NUMBER
2176	

MAIL DATE	DELIVERY MODE
04/03/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 10/665,305
Filing Date: September 22, 2003
Appellant(s): GOERING, THOMAS

Gregory Grace
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 03/20/08 appealing from the Office action mailed 08/14/06.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

2005/0080756 A1	Hitchcock et al.	04-2005

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Claims 1-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hitchcock et al., US 2005/0080756 A1, 04/14/05 (filed 09/29/03, Provisional Application filed on 06/04/98).

In reference to claims 1 and 9, Hitchcock teaches a method and system for a universal forms engine allowing data sharing between customizable admissions applications. See abstract and page 1, paragraph [0008]-[0012]. Compare to ***"a computer system for generating output modules in a form-based application runtime environment"***. Hitchcock discloses the following features:

-A form engine that permits the creation and processing of customizable electronic forms and selective sharing of information between customized forms. A user enters data only once, and the data is shared through an extensible database between disparate forms. The forms engine presents a form to a user for input, receives data from the user, provides the data to the appropriate entity. The forms engine integrates the form and the data. User information and application information are abstracted from the coding, that is the user information and application information is stored in a way that allows the application information and user information to be changed without reprogramming. This abstraction allows a set of user data to be extended without reprogramming, allows user data to be displayed in different formats, and allows the data to be validated. See page 1, paragraphs [0011]-[00015]. The forms engine uses the application data file to produce the requested application in HTML format for display. The application description file can be easily modified, for example to change labels or to add additional fields without reprogramming the forms engine. See page 5, paragraph [0065]. The application data file can be instructed to override default values and can be customized. See page 6, paragraph [0080]. Compare to ***"a form manager component configured to receive an indication that a reusable form element has***

been changed, determine which of the output modules from a set of output modules are affected by the changed form element".

-Creating forms, parsing data on the forms, storing data, retrieving data, and deploying data onto other forms. New forms are automatically populated with the previously entered data. See page 1, paragraph [0012]. The applicant database can be extended to include new attributes without making any changes to the forms engine program or to the application files of institutions that chose not to include the new data. The forms engine automatically uses the application data file to produce the requested application in HTML format for display on the applicant's browser. The application description file can be easily modified, for example, to change labels or to add additional fields. The appearance of the application for each institution can be changed by changing its application description file, without reprogramming the forms engine. The completed application is transmitted to the institution with the data in any format that the institution prefers. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065]. Compare to ***"a runtime manager component configured to receive a request for an output module from the set of output modules and cause regeneration of the requested output module"***.

The claimed “invalidate” step is a means to indicate that the current form (output module) is not valid because of the element change. Hitchcock does not expressly state the output module is invalidated; however, he does teach that as the user enters or customizes data only once, and the data is shared through an extensible database between disparate forms. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065].

Hitchcock further teaches that when an applicant subsequently applies to a different institution, a new application, customized for the different institution is presented to the applicant. Information that was entered into the previously submitted applications is retrieved from the database and presented to the applicant as populated fields of the new application, so that the applicant is not required to enter information more than once. The applicant can, however, change the values in the pre-populated field if desired and the new values are saved for use in subsequent applications. See page 4, paragraph [0056]. The entry of new values in the pre-populated field invalidates the pre-populated fields throughout the other applications because the new values are now being used. Thus the step of “invalidating” is simply a means of “noting” or “acknowledging” that the current data is not to be used. Thus by overriding the pre-populated data in the subsequent applications, Hitchcock’s system is “noting” that the old data is no longer “valid”.

It would have been obvious to a person of ordinary skill in the art at the time of the invention that Hitchcock's ability to merge information and data changes to other forms would entail invalidating other related forms containing incorrect data because the Hitchcock's system is equipped with the ability to "share" data among common application elements (i.e. address info, name) in order to cut down on redundancy and avoid the additional expense and errors of re-keyboarding information in multiple forms having the same data. See page 5, paragraph [0065] and page 1, paragraphs [0003]-[0006]. The entry of new values in the pre-populated field invalidates the pre-populated fields throughout the other applications because the new values are now being used. Thus the step of "invalidating" is simply a means of "noting" or "acknowledging" that the current data is not to be used. Thus by overriding the pre-populated data in the subsequent applications, Hitchcock's system is "noting" that the old data is no longer "valid".

In reference to claims 2 and 10, Hitchcock that after the applicant completes an application for one institution, the data is saved in a database and automatically populates fields in subsequent application forms. See abstract.

In reference to claims 3 and 11, Hitchcock teaches the Application Data File is a specially formatted text file that acts as an application description. It is a series of "directives" and optional arguments which the forms engine parses to build the HTML form and to merge in user data. The directives are interpreted by means of a look-up in a data structure that stores the directive interpretations. See page 6, paragraph [0080].

In reference to claims 4 and 12, Hitchcock does not expressly state the output module is invalidated by marking a flag associated with the module; however, Hitchcock further teaches validating the entire set of data to ensure the data fields are complete. If a field is incomplete, then the field will be flagged during validation. See page 8, paragraph [0117].

In reference to claims 5 and 13, Hitchcock teaches creating forms, parsing data on the forms, storing data, retrieving data, and deploying data onto other forms. New forms are automatically populated with the previously entered data. See page 1, paragraph [0012]. The applicant database can be extended to include new attributes without making any changes to the forms engine program or to the application files of institutions that chose not to include the new data. The forms engine automatically uses the application data file to produce the requested application in HTML format for display on the applicant's browser. The application description file can be easily modified, for example, to change labels or to add additional fields. The appearance of the application for each institution can be changed by changing its application description file, without reprogramming the forms engine. The completed application is transmitted to the institution with the data in any format that the institution prefers. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. This step would entail identifying those forms for which the changes are to be merged. See page 5, paragraph [0065].

In reference to claims 6 and 14, Hitchcock teaches most institutions have application date windows during which applications, whether electronic or paper, for a particular term are accepted. The forms engine verifies that the application is being submitted within the allowed window. Unlike pre-printed paper applications, however, the invention provides the schools the flexibility of easily changing the application date window, so that the time to apply can be extended if the institution wants to receive additional applications. Forms engine uses data from the appropriate application data file (FIG. 14) and previously entered user data to generate a page of a form. See page 4, paragraphs [0053]-[0054].

In reference to claims 7 and 15, Hitchcock disclose a template file gives the application developer absolute freedom to quickly update the application with no need to rewrite or add program code to the forms engine. Use of templates also dramatically reduces the number of functions needed by the engine, as well as the execution overhead. The template file can be in the form of specially tagged HTML; that is, instead of a line-by-line set of directives, the template can look like HTML with embedded special tags representing the form element/variable/value to interpolate. To process the template, the forms engine need only look for <QUESTION> . . . </QUESTION> sections and parse them. Many other pieces of logic could also be embedded into the templates.

In reference to claims 8 and 16, Hitchcock teaches the application description file can be easily modified, for example, to change labels or to add additional fields. The appearance of the application for each institution can be changed by changing its application description file, without reprogramming the forms engine. The completed application is transmitted to the institution with the data in any format that the institution prefers. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065].

In reference to claims 17-18, Hitchcock teaches an applicant database that can be extended to include new attributes without making any changes to the forms engine program or to the application files of institutions that chose not to include the new data. The forms engine automatically uses the application data file to produce the requested application in HTML format for display on the applicant's browser. The application description file can be easily modified, for example, to change labels or to add additional fields. The appearance of the application for each institution can be changed by changing its application description file, without reprogramming the forms engine. The completed application is transmitted to the institution with the data in any format that the institution prefers. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065]. Hitchcock teaches validating the entire set of data to ensure the data fields are complete. If a field is incomplete, then the field will be flagged during validation. See page 8, paragraph [0117]. Compare to ***"responsive to a call to start a form output process based on an identified form: determining whether a previously generated output module associated with the identified form in the output module library has been marked as invalid; if so: regenerating the output module;***

Hitchcock does not expressly state the output module is marked as invalid in a library; however, he does teach that as the user enters or customizes data only once, and the data is shared through an extensible database between disparate forms. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065]. Hitchcock further teaches that when an applicant subsequently applies to a different institution, a new application, customized for the different institution is presented to the applicant. Information that was entered into the previously submitted applications is retrieved from the database and presented to the applicant as populated fields of the new application, so that the applicant is not required to enter information more than once. The applicant can, however, change the values in the pre-populated field if desired and the new values are saved for use in subsequent applications. See page 4, paragraph [0056]. The entry of new values in the pre-populated field invalidates the pre-populated fields throughout the other applications because the new values are now being used. Thus the step of "invalidating" is simply a means of "noting" or "acknowledging" that the current data is not to be used. Thus by overriding the pre-populated data in the subsequent applications, Hitchcock's system is "noting" that the old data is no longer "valid".

Hitchcock does not teach ***"storing the regenerated output module in the output module library along with a marker to indicate that the output module is valid"***. A library is a collection of documents (or output modules). Hitchcock teaches storing forms in an application system database (same as library). The claimed "invalidate" step is a means to indicate that the current form (output module) is not valid because of the element change.

Further, by flagging an incomplete field or field that does not meet the rules specified by an institution for a particular application, Hitchock is marking the output module as invalid and by inputting the new values and validating it as complying with the rules specified by the institution, he is marking it as valid. See page 8, paragraph [0117].

It would have been obvious to a person of ordinary skill in the art at the time of the invention that Hitchcock's ability to merge information and data changes to other forms would entail marking and invalidating other related forms containing incorrect data because the Hitchcock's system is equipped with the ability to "share" data among common application elements (i.e. address info, name) in order to cut down on redundancy and avoid the additional expense and errors of re-keyboarding information in multiple forms having the same data. See page 5, paragraph [0065] and page 1, paragraphs [0003]-[0006]. Further, by flagging an incomplete field or field that does not meet the rules specified by an institution for a particular application, Hitchock is marking the output module as invalid and by inputting the new values and validating it as complying with the rules specified by the institution, he is marking it as valid. See page 8, paragraph [0117].

(10) Response to Argument

On pages 4-5 of the Brief, Appellant argues Hitchcock fails to teach or suggest invalidation of output modules. Specifically, Appellant argues Hitchcock teaches against invalidating output modules by stating that the applicant database can be extended to include new attributes without making any changes to the forms engine program. Appellant argues that because invalidation would lead to changes to the forms engine and Hitchcock cannot be reasonably cited to teach invalidation of output modules.

Examiner disagrees.

The claimed “invalidate” step is simply a means to indicate that the current form (output module) is not valid because of the element change. It is further noted that the invalidating step does not require there to be a “tangible” indication that an output module is invalid. In other words, the “invalidating” step is abstract. Thus, in the previous office actions, Examiner’s position indicated that although Hitchcock does not expressly state the output module is invalidated, he does teach that as the user enters or customizes data, the data is shared through an extensible database between disparate forms. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. See page 5, paragraph [0065].

Hitchcock further teaches that when an applicant subsequently applies to a different institution, a new application, customized for the different institution is presented to the applicant. Information that was entered into the previously submitted applications is retrieved from the database and presented to the applicant as populated fields of the new application, so that the applicant is not required to enter information more than once. The applicant can, however, change the values in the pre-populated field if desired and the new values are saved for use in subsequent applications. See page 4, paragraph [0056]. It is the Examiner’s view that the **entry of new values in the pre-populated field invalidates the pre-populated fields throughout the other applications because the new values are now being used**. Thus the step of

“invalidating” is simply a means of “noting” or “acknowledging” that the current data is not to be used. Thus by overriding the pre-populated data in the subsequent applications, Hitchcock’s system is “noting” that the old data is no longer “valid”.

In response to the comments above, Appellant argues on page 5 of the Brief that Examiner’s position is in error because the meaning of an “output module” is a module that provides output, not the output itself. Appellant states that a form output module process form data for presentation which is different from form data.

Examiner disagrees that Hitchcock only teaches invalidating form data and not the output module.

If an output module is responsible for outputting data for presentation and the data has been altered in a manner that affects data in other forms, then the output module is "invalidated". The **entry of new values in the pre-populated field invalidates the pre-populated fields throughout the other applications and forms because the new values are now being used**. This entry of new values “invalidates” the form output module because it cannot output the form using the old values. In fact the output module can only present the new data and the output modules that would have presented the old data are no longer able to output the old data.

As further evidence of this fact, Examiner refers to page 2, lines 14-24 of the Brief where Appellant states, *“The system and methods of claims 1, 9, and 17 enable form elements to be shared by multiple forms and for the changes to the form element to trigger the regeneration of the output module that use the changed form element in* *order to incorporate the changes (Specification, paragraph 8).* In other words, as stated in the paragraph above, a change to a data element does affect the output module in that it invalidates the module because the data element has been altered. Furthermore, it is noted that the paragraph (paragraph 8) relied upon by Appellant as defining output module does not appear to be accurate.

On pages 6-7, Appellant argues Hitchock fails to teach regenerating invalidated output modules.

Examiner disagrees.

Hitchcock teaches new forms are automatically populated with the previously entered data. See page 1, paragraph [0012]. The applicant database can be extended to include new attributes without making any changes to the forms engine program or to the application files of institutions that chose not to include the new data. The forms engine automatically uses the application data file to produce the requested application in HTML format for display on the applicant's browser. The application description file can be easily modified, for example, to change labels or to add additional fields. The appearance of the application for each institution can be changed by changing its application description file, without reprogramming the forms engine. The completed application is transmitted to the institution with the data in any format that the institution prefers. The institution can therefore upload the data directly into its applicant or student information system database, merging the information seamlessly into their existing workflow, thereby avoiding the additional expense and errors of re-keyboarding the information. The forms engine thus has the capability of outputting application information universally across platforms. See page 5, paragraph [0065].

Further, it is noted that the invalidating step does not require there to be a "tangible" indication that an output module is invalid. In other words, the "invalidating" step is abstract.

In response to the comments above, Appellant argues on pages 7-8 that the new forms being populated with new data is not regenerating an output module.

Examiner notes that changes to data result in a different presentation. Thus the output module must be regenerated if the data elements have been altered. As stated above, this is further evidenced by Appellant's remarks on page 2, lines 14-24 of the Brief where Appellant states, *"The system and methods of claims 1, 9, and 17 enable form elements to be shared by multiple forms and for the changes to the form element to trigger the regeneration of the output module that use the changed form element in order to incorporate the changes (Specification, paragraph 8).* In other words, as stated in the paragraph above, a change to a data element does affect the output module in that it invalidates the module because the data element has been altered. In order to present the form with new data, a new output module is generated with the changes to the data elements.

Regarding dependent claims 2-8, 10-16, and 18, Appellant relies on previous arguments above with respect to these claims. Accordingly, the arguments detailed above are relied upon with respect to dependent claims 2-8, 10-16, and 18.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Rachna Singh
/Rachna Singh/
Primary Examiner, Art Unit 2176

March 31, 2008

Conferees:

Doug Hutton

/Doug Hutton/
Doug Hutton
Supervisory Primary Examiner
Technology Center 2100

William Bashore

/William L. Bashore/
William L. Bashore
Primary Examiner
Tech Center 2100